A New Approach to Global Minimization

AARON F. STANTON,[1] RICHARD E. BLEIL,[2] and SABRE KAIS[1]*

[1]*Department of Chemistry, Purdue University, West Lafayette, IN 47907; and* [2]*Kettering College of Medical Arts, 3737 Southern Boulevard, Kettering, OH 45429*

Received 19 April 1996; accepted 26 July 1996

ABSTRACT

A new algorithm is presented for the location of the global minimum of a multiple minima problem. It begins with a series of randomly placed probes in phase space, and then uses an iterative Gaussian redistribution of the worst probes into better regions of phase space until all probes converge to a single point. The method quickly converges, does not require derivatives, and is resistant to becoming trapped in local minima. Comparison of this algorithm with others using a standard test suite demonstrates that the number of function calls has been decreased conservatively by a factor of about three with the same degree of accuracy. A sample problem of a system of seven Lennard–Jones particles is presented as a concrete example.

Introduction

The recent explosion of publications relating algorithms for the location of the global minimum in a multiple minima function demonstrates both the importance of this problem in practically all fields of science and that further development toward such a method is still needed. Such an algorithm would find applications in fields such as drug design, molecular modeling, quantum mechanical calculations, and mathematical biological calculations.[1–8] Unfortunately, the problem of locating the global minimum in nearly all cases is hindered by the presence of local minima.

All minimization algorithms face the difficulty of locating the global minimum in the presence of a series of local minima. Once a local minimum is converged on, there is no sure method to determine if another deeper minimum might exist somewhere else without leaving the comfort of the minimum located. The problem becomes more harrowing as one increases the dimensionality of the problem. Fortunately, several newer methods[9] for function minimization are showing vast improvements in the ability to spot local versus global minima.

Some of the older minimization algorithms are local deterministic methods, primarily based on extensions of Newton's methods.[10] These work by following the terrain of the function, either utilizing the numerical steepest downhill approach, or some derivative of the function which indicates the best path to follow. These methods were notorious for finding themselves locked in local minima. More recently, these approaches have been usurped by other methods resistant to becoming

*To whom all correspondence should be addressed. E-mail: sabre@salam.chem.purdue.edu

CCC 0192-8651 / 97 / 040594-06

trapped in local minima. Simulated annealing[11] is based on a classical dynamics approach, wherein a probe is moved and allowed to overcome small maxima in the beginning, thereby having the ability to move out of local minima. The tabu search[12] utilizes several randomly chosen guesses for the global minimum within a given region of phase space, restricting the number of times it can look in that region before it must move on, hopefully to find a lower minimum in another such region. The genetic algorithm[13] also starts with several random guesses as to where the minimum might be, but then changes some of the variables of the worst guesses until they all converge. These methods have been compared in several review articles,[12] and show improving accuracy and efficiency. It must also be noted that the presented method combines features of the genetic algorithm and simulated annealing, similar to work done by Fox[14]; however, this method does not borrow from the tabu search algorithm.

The algorithm being introduced shows a great improvement in efficiency of the location of the global minimum. It begins with a series of random guesses as to where the minimum might be, then redistributes the worst guesses into better regions of phase space until all probes converge to a single point. This new method locates the global minimum in a multiple minima problem, without the necessity of including computationally expensive estimates for the derivative of the function, is generally applicable, avoids entrapment in local minima, and is conceptually simple and easy to program.

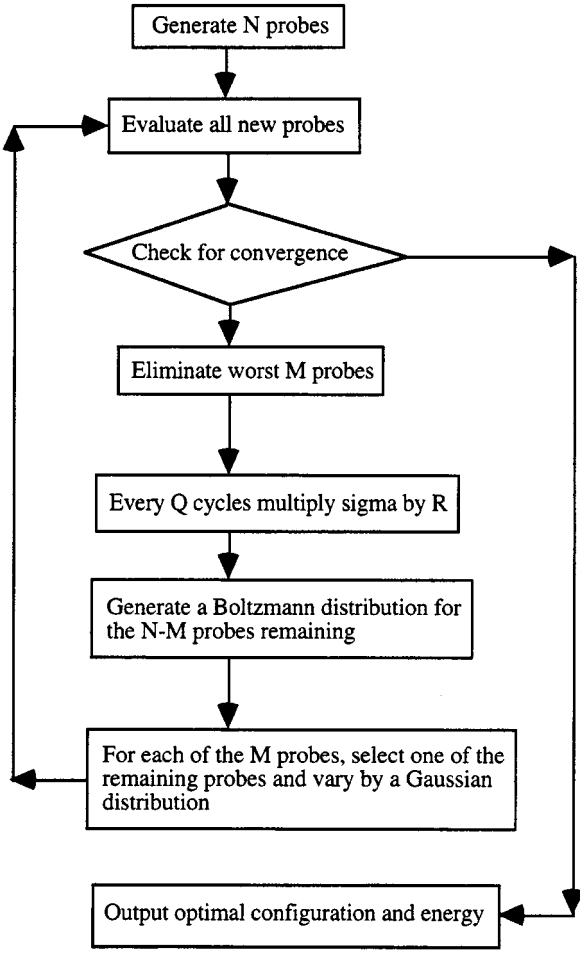

FIGURE 1. Flow chart for this method for global optimization. N is the total number of probes, M is the number of probes to be eliminated, Q is the number of cycles between Gaussian contractions, and R is the Gaussian scaling factor.

Methodology

Phase space is defined to be the set of points bounded by the minimum and maximum values of all variables in the system. The dimensionality of the problem is this number of variables. The task is to find the global minimum within the defined phase space, min $f(s)$: $s \in S$, where $f(s)$ is the function to be minimized and s is a point in phase space S. This begins with the random placement of probes within phase space, which are used as a means of sampling regions of the entire space. For convenience, a flowchart (Fig. 1) has been included to illustrate this method step by step.

The probes are themselves complete sets of all variables in the phase space, and thus each probe has the specific value of the function to be minimized at that location. Initially, these probes are randomly placed within phase space, and the function value at the location of each probe is determined. The number of probes chosen is arbitrary, but with enough probes to get a good statistical distribution. With too many probes, the algorithm runs too slowly, but without enough probes, one is likely to find a local rather than global minimum. A balance between these two extrema is required, with more probes necessary for more sharply varying functions. Once the probes are initially placed, then one begins sampling even more of the phase space by the relocation of the worst of these probes to be nearer to the best.

Before one can choose where to relocate the worst probes, it first becomes necessary to choose how many probes should be relocated. It is then

necessary to rank the existing probes according to their values of the function to be minimized. It must be noted that this method borrows conceptually from the genetic algorithm in this part, but that it substantially differs in how new probes are chosen, as will be detailed. The number of probes to be relocated is also arbitrary. A satisfactory amount is roughly one third of the total number of probes. At each iteration of the algorithm, one then replaces a certain number of the probes, choosing to replace the worst guesses at all times, keeping the best guesses unchanged. The larger the percentage of probes to be relocated, the more calculations that are required at each iteration, but fewer iterations should be required. This saves computational time on reranking the probes at each step. Furthermore, it is necessary to keep enough of the best probes to avoid the possibility that one is deleting the probe closest to the global minimum because a second probe happens to be sitting lower in a local minima well. The probes to be relocated are placed near one of the better probes, called a pivot probe, in a random fashion.

Recognizing the problem that one may be too close to a local minimum in the first iteration, one chooses the pivot probe in a random fashion with a probability based on the value of each probe. To do this, one assigns a probability to each of the probes not to be relocated in a Boltzmann distribution; that is, for probe i, its probability is $P_i = \exp(-f(i))/P$, where $P = \sum_{i=1}^{n-m} \exp(-f(i))$, n is the total number of probes, m is the number of probes relocated at each iteration, and $f(i)$ is the function value of probe i. Each probe to be relocated is placed randomly near one of these remaining probes, with the best probe getting most of the relocated probes near it, but it is important to note that all remaining probes have a finite chance of being used as a pivot probe. Each probe to be relocated chooses its own pivot probe based on these probabilities. Typically, in a single iteration, there are at least three or four pivot probes chosen, depending on how sharply varying the function is, how many choices of pivot probes there are, and how many probes will be relocated. In this way, one is always moving probes to better regions of phase space, but hopefully slowly enough so as not to overlook a global minimum. The distance of the location of the relocated probe from the pivot probe is a decision that must be carefully weighed.

Much like simulated annealing, it is best to begin with a fairly large distribution of relocated probes from the pivot probe. The relocated probes are placed in a Gaussian distribution centered on the pivot probes with a standard deviation input as an arbitrarily adjustable parameter, σ. This standard deviation defines a space around each pivot probe, where the relocated probes are found inside this region half of the time, and outside of this region the rest of the time. Initially, this size should be chosen to be large, much like simulated annealing, so that a fairly large portion of phase space is covered by the pivot probes. This process repeats for several iterations.

The number of iterations for any given standard deviation value can be varied. The more steps taken, the better phase space is sampled, but the longer it will take for the algorithm to converge. Once a predetermined number of iterations have elapsed using this standard deviation, this value is then decreased at some given rate. This rate, R, decreases the standard deviation σ to some new value, σ', where $\sigma' = R\sigma$. The rate used for the functions tested was $R = 0.466$. At each new standard deviation, the algorithm repeats as many times as with the initial standard deviation, until σ is decreased again. In effect, this means that the probes will in time converge on some given small point, and, with a good sampling of phase space, this point should be the global minimum. The choice of stopping criteria therefore becomes important.

There are several possible choices of stopping criteria. One is to choose a small given range of values for the function, or some given standard deviation of the average values of the parameters of phase space. By choosing a stopping criteria based on the standard deviation of the phase space variables, hopefully one avoids finding several local minima wells with equal energies. Unfortunately, it is also possible for the system to become trapped in several minima and, if the cooling rate is too high, to simply never leave those wells, effectively resulting in an infinite loop. It is always a good idea to set a maximum number of iterations to avoid such a loop. In the end, however, it is common practice just to look at the value of the best probe, with the hope that the probe is closest to the true global minimum. If a value for the global minimum is known one can stop once the best probe is within a given error of this value. Of course, the implementation of this algorithm depends on what is known of the function initially.

If the behavior of the function is known to be smoothly varying, it would be best to choose few probes and a large number of probes to be relocated at each iteration. If it is sharply varying, it is best to use a larger number of probes with a

smaller number to be relocated at each iteration. If the behavior of the function is not known at all, it is best to repeat the algorithm at least twice, increasing the number of probes. If a different minimum is found, then it is a sharply varying function and yet another run would be warranted. For the test cases, the values of the global minima were known and we chose to use stopping criteria appropriate to this knowledge.

In the test cases, several well established functions were used for comparison with established methods of optimization. These functions included Goldstein–Price (GP), Branin (BR), Hartman three- and six-dimensional variants (H3 and H6), and Shubert (SH). The full details of the functions tested can be found elsewhere.[12,15] Notice that all of these functions except for the H3 and the H6 are two dimensional and that all of these have similar results. The stopping criteria selected was for the probe with the best value to be at worst within 3% of the known global minimum, or for it to stop if the number of iterations exceeded a certain amount. This latter criteria was set high enough that if it were the cause for stopping the algorithm it could be safely stated that the probes were trapped in a local minima. For the H6 function, repeating boundary conditions were introduced, which essentially cause the probes to ''wrap around'' the area being searched rather than be pushed to the edges of phase space. It was determined that this enabled more of phase space to be searched, and significantly improved the results. Note that this technique was not used on the other functions tested, but the results obtained from H6 strongly indicate that this would not detract in any way from their efficiency and should in fact improve the algorithm overall.

As a rule of thumb, a number of probes equal to 5–15 times the dimensionality of the function were used, and a third of them were moved at each iteration. Clearly, with fewer probes the method runs much more quickly, but one also has a greater chance of being caught in a local minimum. The selected cooling rate came from some preliminary results in attempting to optimize the search parameters and was used throughout for consistency. The initial phase was 10 iterations, as was the number of steps between each reduction of the standard deviation as described above. In our specific example in the next section, we demonstrate that it can be useful to vary the number of steps between cooling. The step size taken is crucial. It is very important that the steps taken are not too small initially, as that will cause probes to never leave the region of a local minima. It is advisable to begin with large steps and cool to much smaller ones. In general, the initial step size should be on the same order of magnitude as the phase space being searched. This especially holds true if one is using repeating boundary conditions, as this insures that all of phase space will be adequately searched. As a specific example, for H6, which has a range of 0–1 for each variable, an initial standard deviation of 1 was used, which implies that half of the time the probe was relocated in such a way that repeating boundary conditions were invoked. This in turn shows that the entire space is searched.

A Concrete Example

The previous sampling of the performance of this new algorithm using a standard test suite of mathematical functions suffices to demonstrate its usefulness for functions of low dimensionality. In general, however, one is interested in a function of more than only six dimensions. Therefore, to illustrate this method's usefulness to more realistic problems, we present a simple Lennard–Jones cluster of seven particles. While this is clearly very far from anything new, its very simplicity can be used to implement this method relatively easily.

Specifically, one must establish an array containing the coordinates of seven points in three-dimensional space, and a function that yields the standard Lennard–Jones energy when that array is passed to it.[8] One must also establish an array capable of holding an arbitrary number of these sets of seven points. Each set of seven is referred to as a ''probe'' of the space in question. In our particular implementation we chose to use 150 of these ''probes'' to explore the potential.

One must then place each of these ''probes'' in the space to be searched. We chose to confine our search to a region of three-dimensional space defined by a cube extending from -2 to 2 in each direction, and any ''probe'' which moved outside of this region was placed back into it by using repeating boundary conditions. Each ''probe'' is placed randomly, which is to say that for each of our 150 ''probes'' we randomly place seven points inside the cube from -2 to 2.

Next, each ''probe'' has an energy assigned to it by using the Lennard–Jones function previously mentioned, and the whole set of 150 is sorted from best to worst (lowest to highest). Specifically, we use the Quicksort algorithm for this purpose.

At this point, we chose to discard the worst 50 sets of points and choose 50 new sets of points based on the remaining 100. Each of these ''probes'' is assigned a probability of being chosen according to a Boltzmann distribution, as explicitly described in the previous section. Each of the 50 new ''probes'' then separately chooses a ''probe'' to be based on. Note that it is entirely possible for two or more of the new sets of points to be based on the same ''probe.'' We then vary the selected set of points by a Gaussian distribution around it, and the new values of coordinates for the set of points becomes the new ''probe.'' We initially chose for the width of this Gaussian distribution to be 2, or one half the width of the cube being searched. This serves to establish that all of the space is being searched properly.

At this point, all 50 of the new ''probes'' must have their energies evaluated, and then the entire set of 150 is resorted. Again, the worst 50 are selected out, rechosen, reevaluated, and reranked. This entire process continues 100 times in our particular example.

After this occurs 100 times, we change the width of the Gaussian distribution by a factor of 0.9, which serves to narrow the region of space being searched. We wish to confine our search to regions that we believe likely to yield the global minimum. At this point we also determined the statistical deviation of the energies of all 150 probes. If the deviation is below 10^{-7} we decide that the system has converged sufficiently to assume that we are finished. If the deviation is insufficiently low we repeat another 100 steps of selecting out and choosing new ''probes,'' and at the end of this time the Gaussian is once again contracted and statistics are once again performed.

Clearly, we could have used the known global minimum value of -16.505[8] for this particular problem as the stopping criteria, but we wish to demonstrate that this method is fully capable of finding an unknown global minimum quite as effectively as a known one. Having run this entire routine 100 times, this method found the global minimum of -16.505 a total of 75 times. In addition, it took this method an average of 390,383 calls to the potential for each of the times that it successfully found that minimum. Admittedly, this does not represent a 90% convergence, but it suffices to prove that this method will find a global minimum without previously knowing exactly what that number is. One can improve the convergence rate if one is willing to sacrifice speed in so doing simply by increasing the number of probes or by increasing the number of cycles between cooling.

This simple demonstration of this method is quite easily repeated in any programming language capable of performing simple mathematical operations. One run of this particular implementation takes approximately 7–10 minutes on a 486DX/33 with 8 megabytes of RAM, and takes substantially less time on an IBM RS/6000. One hundred runs of this took less than 9 hours on a nondedicated RS/6000. This should demonstrate adequately that this method is quite fast.

The bulk of the time is in merely insuring that the implementation itself is behaving properly, and with this simple illustration one should be able to follow along with little trouble.

Results and Discussion

The results given in Table I reflect this algorithm converging a minimum of 90% of the time to within, at most, 3% of the known global minimum of the function in question. In many cases the error was significantly less than 3%, and the convergence exceeds 90% on a few. This is identical with the established reliability of the tabu search method.[12] The number of function calls listed is the average value of the times that the algorithm converged. This indicates a two- to fourfold improvement over the best method found for optimization, the tabu search method, for all functions tested. It must be noted, however, that Kan and Timmer[16] have released results that are par-

TABLE I.
Number of Function Evaluations in the Global Optimization of Five Test Functions Defined in the Text.[a]

Method[b]	GP	BR	H3	H6	SH
PRS	5125	4850	5280	18,090	6700
MS	4400	1600	2500	6000	—
SA1	5439	2700	3416	3975	241,215
SA2	563	505	1459	4648	780
TS	486	492	508	2845	727
Present work	112	144	122	1536	281

[a] The majority of these data are taken directly from ref. 12. The results of the new method have been added for comparison.
[b] The methods are the pure random search (PRS), multistart (MS), simulated annealing types 1 and 2 (SA1 and SA2, respectively), and tabu search (TS). The references for these methods can be found in ref. 12.

ticularly noteworthy for a similar test suite of functions, but a lack of clearly staged convergence criteria prevents one from using their results as a basis of comparison. Had such criteria been established, their results would be included in the table to indicate the relative merit of the various methods presented. We also have presented a concrete example that should be very easy for anyone familiar with programming to implement, which enables users to see directly the value of this method.

This method of searching for a global minimum of an arbitrary function has proven to be efficient, seldom gets trapped in a local minimum, does not require the computationally expensive use of derivatives, and is easy to implement. A mathematical basis for the efficiency of this method is currently being investigated, which hopefully will lead to additional refinements. We are currently using this method to find minimum energy structures of noble gas clusters, as well as electronic structure using dimensional scaling theory.[3,17]

References

1. K. B. Lipkowitz and D. B. Boyd, Eds., *Reviews in Computational Chemistry*, *Vol. III*, VCH, New York, 1992.
2. L. C. W. Dixon and G. P. Szego, *Towards Global Optimization*, *Vols. 1 and 2*, North-Holland, New York, 1975.
3. D. R. Herschbach, J. Avery, and O. Goscinski, *Dimensional Scaling in Chemical Physics*, Kluwer, Dordrecht, 1993.
4. T. L. Blundell, B. L. Sibanda, M. J. E. Sternberg, and J. M. Thornton, *Nature*, **326**, 347 (1987).
5. K. A. Dill, S. Bromberg, K. Yue, K. M. Fiebig, D. P. Yee, P. D. Thomas, and H. S. Chan, *Prot. Sci.*, **4**, 561 (1995).
6. M. Oresic and D. Shalloway, *J. Chem. Phys.*, **101**, 9844 (1994).
7. L. Piela, J. Kostrowicki, and H. A. Scheraga, *J. Phys. Chem.*, **93**, 3339 (1989).
8. S. K. Gregurick, M. H. Alexander, and B. Hartke, *J. Chem. Phys.*, **104**, 2684 (1996).
9. C. Floudas and P. M. Pardalos, *Recent Advances in Global Optimization*, Princeton University Press, Princeton, NJ, 1991.
10. M. H. Wright, *Acta Numerica*, **1**, 341 (1991); J. Nocedal, *Acta Numerica*, **1**, 199 (1991).
11. S. Kirkpatrick, C. D. Gelatt Jr., M. P. Vecchi, *Science*, **220**, 671 (1983).
12. D. Cvijovic and J. Klinowski, *Science*, **267**, 664 (1995).
13. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, Reading, MA, 1989.
14. B. L. Fox, *Ann. Oper. Res.*, **41**, 47 (1993).
15. A. Torn and A. Zilniskas, *Global Optimization*, Princeton University Press, Princeton, NJ, 1991.
16. A. Rinooy Kan and G. Timmer, In *Numerical Optimization, 1984*, P. Boggs, R. Byrd, and R. Schnabel, Eds., SIAM, Philadelphia, PA, 1985, p. 245.
17. A. Stanton, R. Bleil, and S. Kais (in preparation).